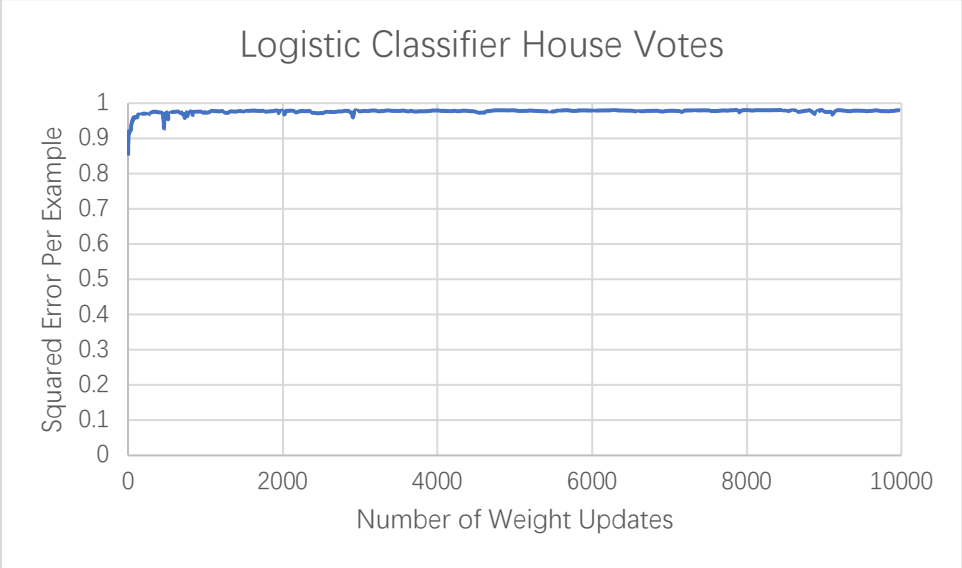
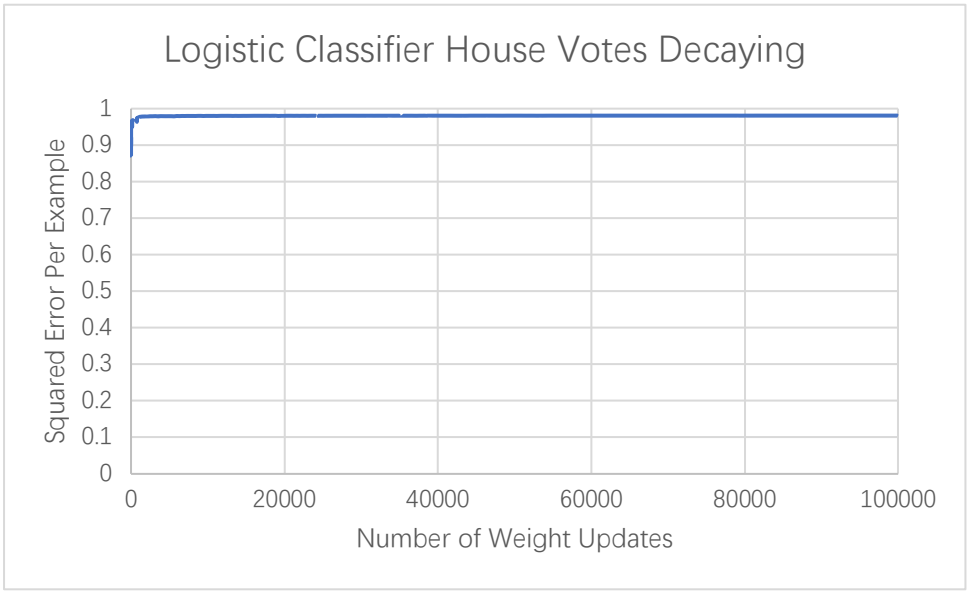
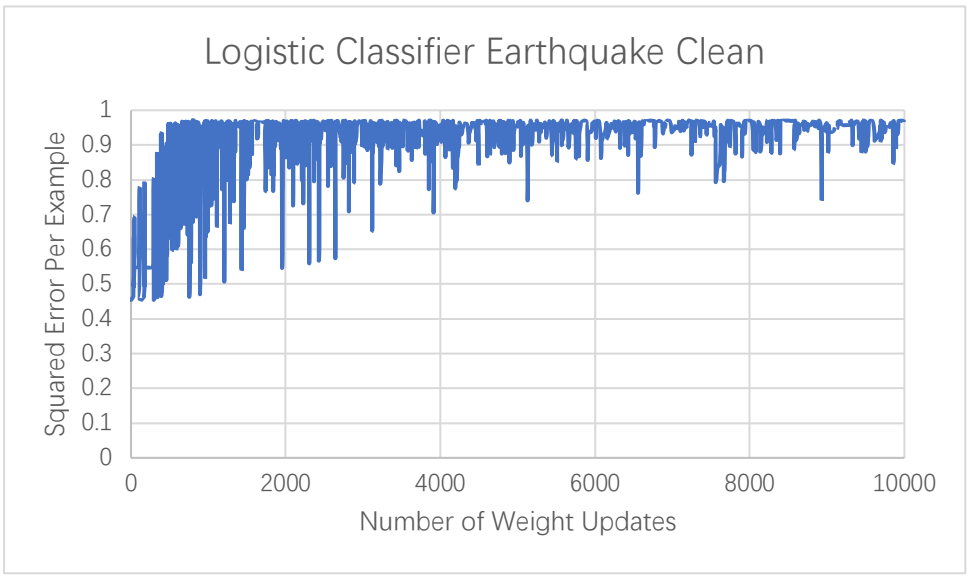
1. How to Run the code in terminal (Please follow the following steps one by one carefully to run the code)

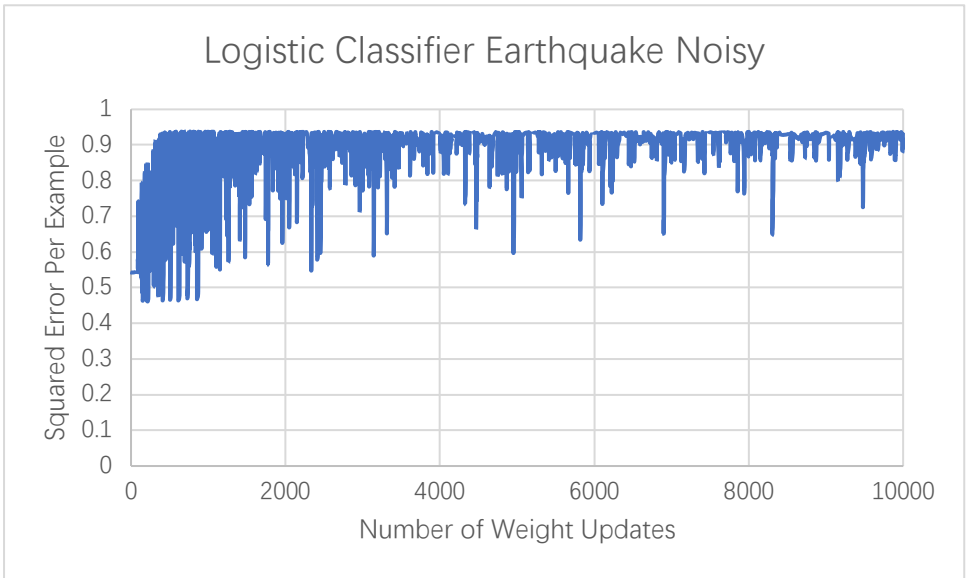
- cd to src directory: **cd src**
- Compile all the classes in the src directory:
  javac learn/lc/core/*.java learn/lc/examples/*.java learn/lc/display/*.java learn/math/util/*.java
- Please enter: "java learn/lc/examples/" before the class name to run classifier
  java learn/lc/examples/[classname]
  - For example: java learn/lc/examples/LogisticClassifierEarthquakeNoisy
  - You can test all the LogsiticClassifier or PerceptronClassifier with this form, so you don't need to enter any parameter. It is same as entering parameter in LogisticClassifierTest and PerceptronClassifierTest.
- If you want to test with display or using classifier test directly. Please use the following format:
  java learn/lc/examples/[classname] [data filename] [step you want] [alpha]
  - For example: java learn/lc/examples/LogisticClassifierTest earthquake-noisy.data.txt 10000 0.95

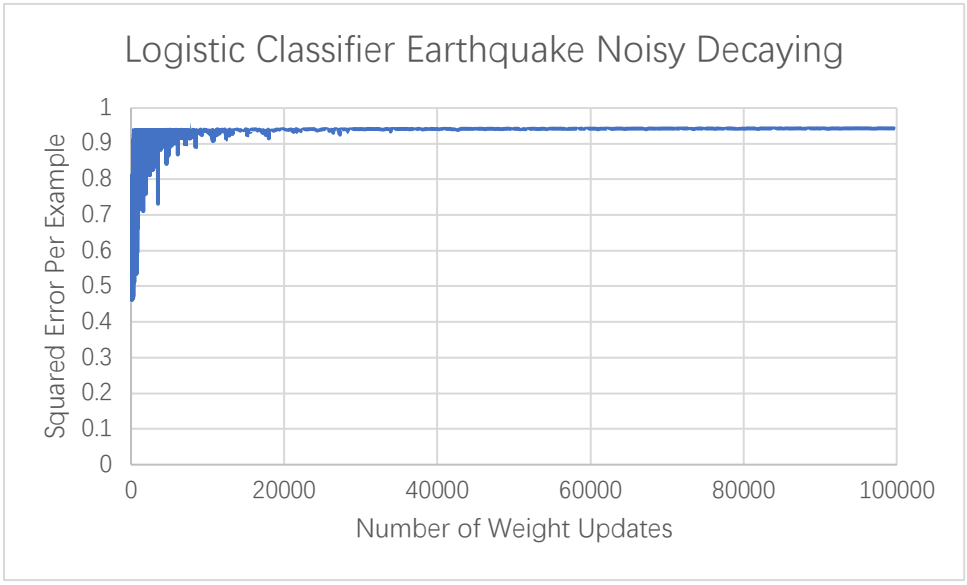
2. Graphs

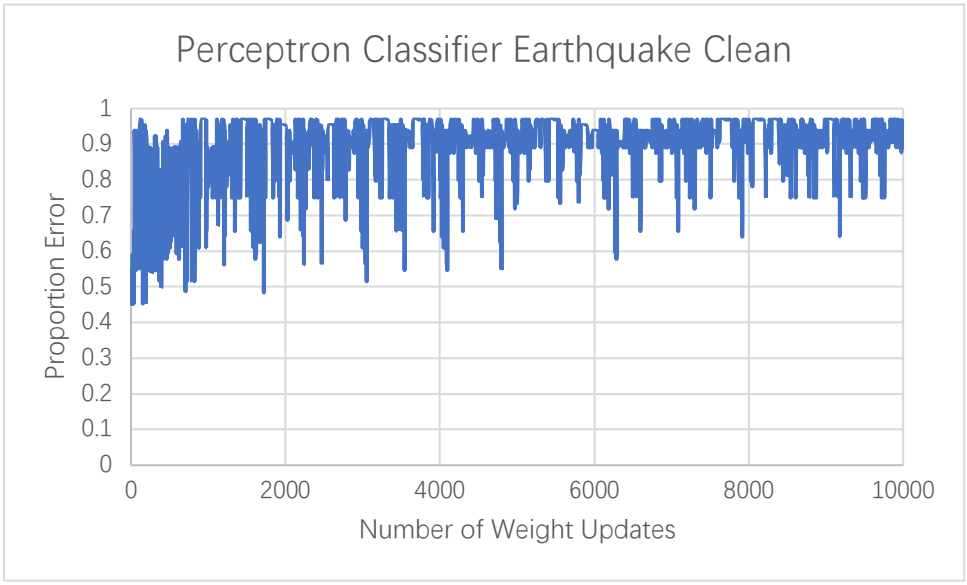


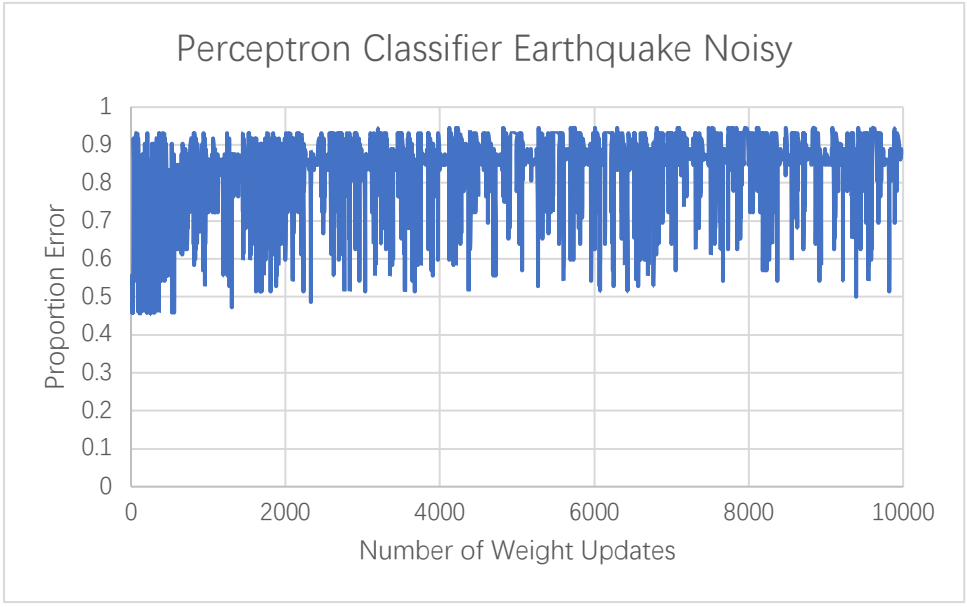

Logistic Classifier House Votes Decaying
Squared Error Per Example
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
0
20000
40000
60000
80000
100000
Number of Weight Updates

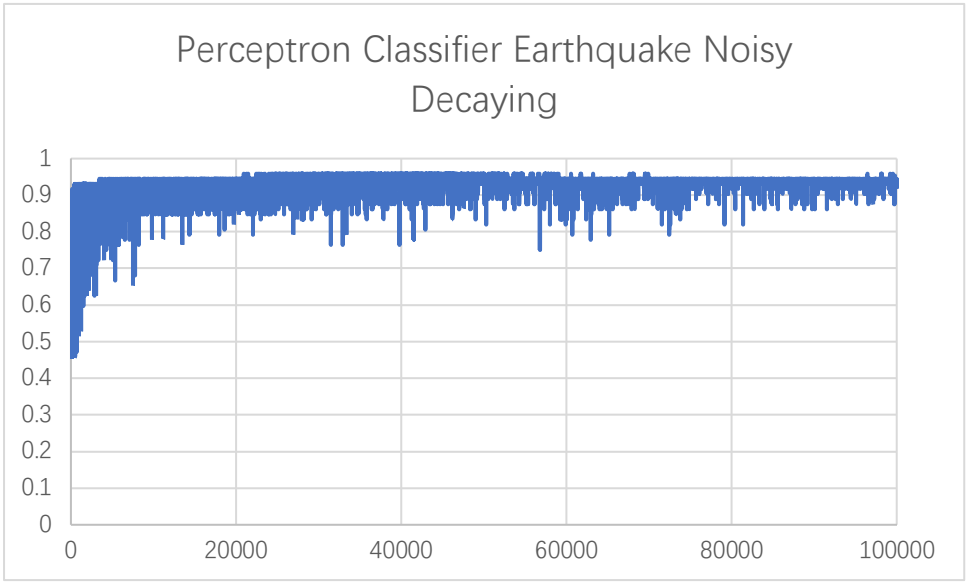

Logistic Classifier Earthquake Clean
Squared Error Per Example
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
0
2000
4000
6000
8000
10000
Number of Weight Updates

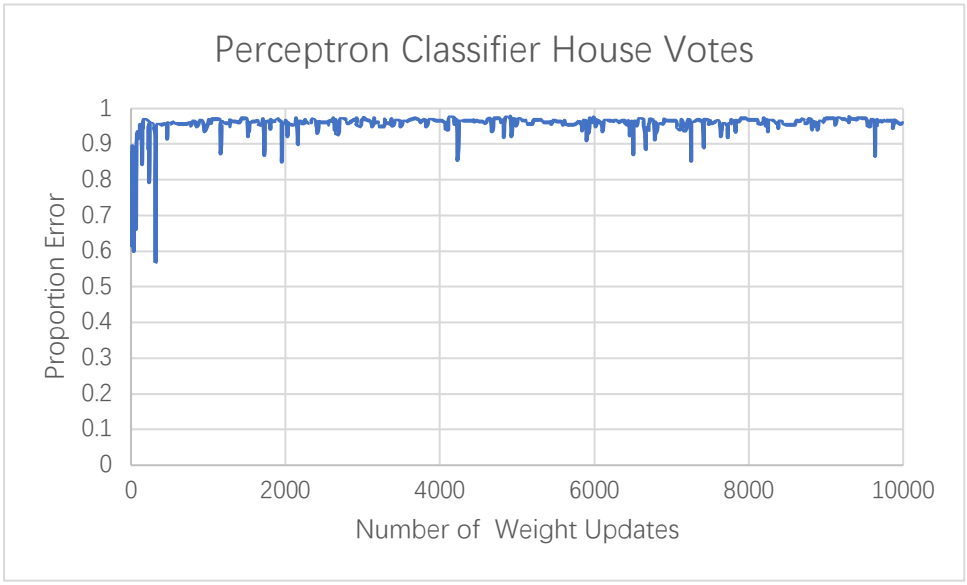

Logistic Classifier Earthquake Noisy
Squared Error Per Example
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
0
2000
4000
6000
8000
10000
Number of Weight Updates

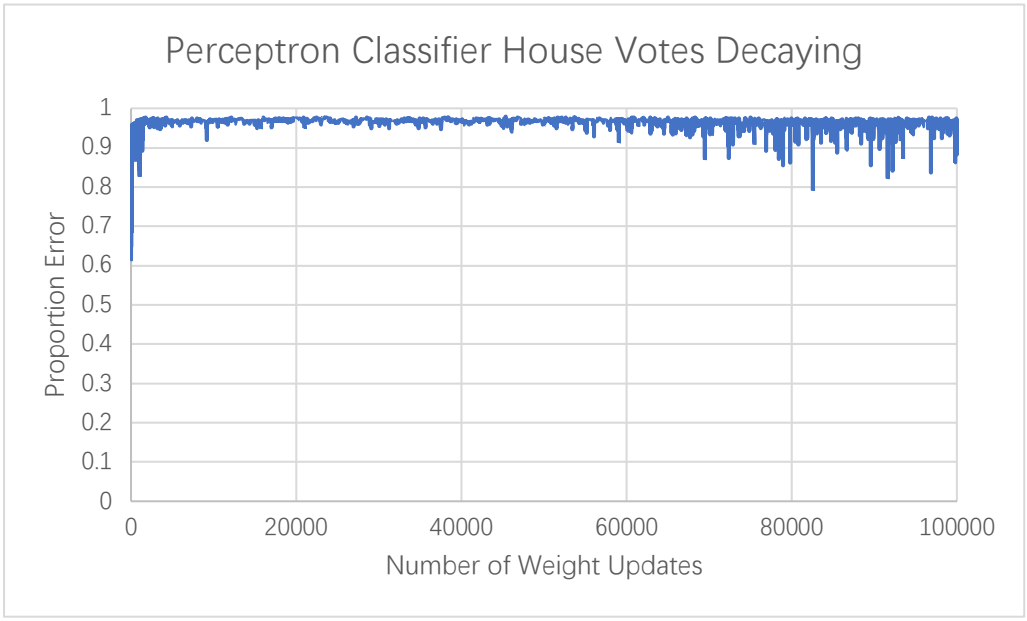

Logistic Classifier Earthquake Noisy Decaying
Squared Error Per Example
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
0
20000
40000
60000
80000
100000
Number of Weight Updates


Perceptron Classifier Earthquake Clean
Proportion Error
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
0
2000
4000
6000
8000
10000
Number of Weight Updates


Perceptron Classifier Earthquake Noisy
Proportion Error
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
0
2000
4000
6000
8000
10000
Number of Weight Updates


Perceptron Classifier Earthquake Noisy Decaying
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
0
20000
40000
60000
80000
100000


Perceptron Classifier House Votes
Proportion Error
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
0
2000
4000
6000
8000
10000
Number of Weight Updates


Perceptron Classifier House Votes Decaying
Proportion Error
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
0
20000
40000
60000
80000
100000
Number of Weight Updates

3. (For Extra Credit) How to run the code of the extra credit part in the terminal. (Please follow the following steps one by one carefully to run the code)

- cd to src directory: **cd src**
- Compile all the classes in the src directory:
  javac learn/nn/core/*.java learn/nn/examples/*.java learn/math/util/*.java
- Please enter: "java learn/nn/examples/" before the class name to run classifier
  java learn/nn/examples/[classname]
  - For example: java learn/nn/examples/IrisNN
  - You can use this format to test both Iris dataset and MNIST dataset

This is we have when we run our code in the terminal (Iris Dataset):

```
RainydeMacBook-Pro:src rainyzhao$ cd
/Users/rainyzhao/Desktop/CSC242Project4/src
RainydeMacBook-Pro:src rainyzhao$ javac learn/nn/core/*.java
learn/nn/examples/*.java learn/math/util/*.java
RainydeMacBook-Pro:src rainyzhao$ java learn/nn/examples/IrisNN
Training for 1000 epochs with alpha=0.1
LAYER UNIT    w_0 ...
1    0    0.02    5.15    5.51    -7.31    -8.80
1    1    -0.03    5.43    5.78    -7.68    -9.24
1    2    -0.01    -0.66    -0.90    1.111.78
1    3    -0.02    0.65    1.87    -3.12    -1.49
1    4    -0.02    0.63    1.69    -2.90    -1.47
1    5    0.02    -0.29    0.18    0.32    1.05
1    6    -0.02    -2.24    -2.32    3.14    4.19
2    0    -0.04    -0.57    -0.58    -2.65    4.03    3.42    -2.61    -2.00
2    1    0.00    4.22    4.56    -0.60    -6.92    -5.85    -1.14    -1.86
2    2    0.03    -5.14    -5.18    1.31    -1.65    -1.84    0.76    1.93
Overall accuracy=0.9666666666666667

Confusion matrix:
      Predicted
Actual  0    1    2
0    1.000    0.000    0.000
1    0.000    0.900    0.100
2    0.000    0.000    1.000

k-Fold Cross-Validation: n=150, k=10
SET      ACCURACY
0        1.000
1        1.000
2        1.000
3        1.000
4        0.733
```

5	0.867
6	1.000
7	1.000
8	0.800
9	1.000

average accuracy: 0.940

Learning Curve testing on all training data

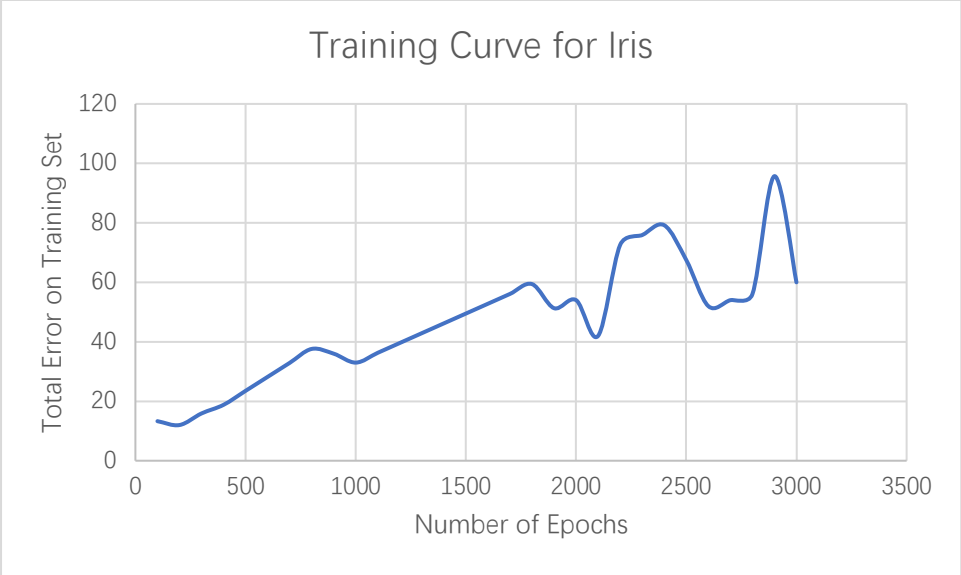
EPOCHS	ACCURACY
100	0.867
200	0.940
300	0.947
400	0.953
500	0.953
600	0.953
700	0.953
800	0.953
900	0.960
1000	0.967
1100	0.967
1200	0.967
1300	0.967
1400	0.967
1500	0.967
1600	0.967
1700	0.967
1800	0.967
1900	0.973
2000	0.973
2100	0.980
2200	0.967
2300	0.967
2400	0.967
2500	0.973
2600	0.980
2700	0.980
2800	0.980
2900	0.967
3000	0.980

This is we have when we run our code in the terminal (MNIST Dataset):

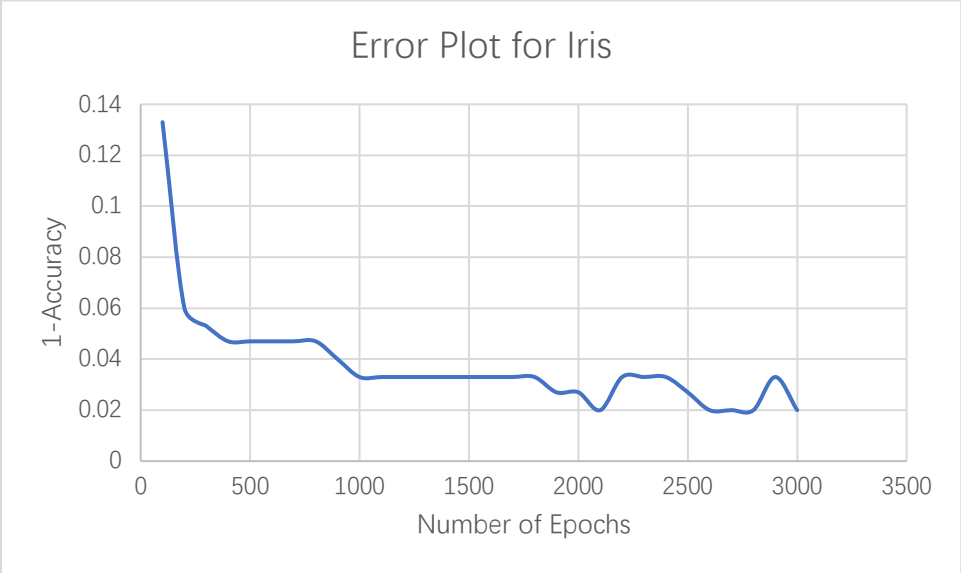
EPOCH	ACC	TIMEms	HHMMSS
1	0.446	809398	00:13:29
2	0.477	798548	00:13:18

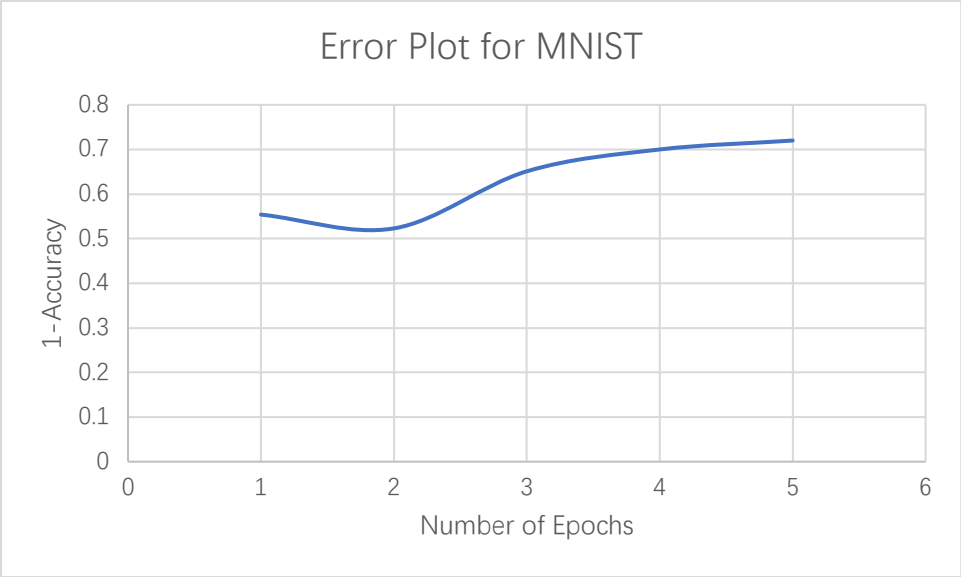
3	0.349	773397	00:12:53
4	0.300	772705	00:12:52
5	0.280	830411	00:13:50

4. Graphs for Extra Credit Part:



(ps: This curve is an inverse graph of Fig 18.25 (2))





(ps: Since testing the examples takes too much time, we only get 5 epochs in 90 minutes. So we can only draw a plot with these 5 points. Because the number of the examples we have are too limited, we cannot draw the training curve.)